1.3 BPF_PROG_TYPE_SK_SKB

- What do I do with it? Allows users to access skb and socket details such as port and IP address with a view to supporting redirect of skbs between sockets. See https://lwn.net/Articles/731133/ . This functionality is used in conjunction with a sockmap – a special-purpose BPF map that contains references to socket structures and associated values. sockmaps are used to support redirection. The program is attached and the bpf_sk_redirect_map() helper can be used to carry out the redirection. The general approach we catch socket creation events with sock_ops BPF programs, associate values with the sockmap for these, and then use data at the sk_skb instrumentation points to inform socket redirection – this is termed the verdict, and the program for this is attached to the sockmap via BPF_SK_SKB_STREAM_VERDICT. The verdict can be __SK_DROP, __SK_PASS, or __SK_REDIRECT. Another use case for this program type is in the strparser framework (https://www.kernel.org/doc/Documentation/networking/strparser.txt). BPF programs can be used to parse message streams via callbacks for read operations, verdict and read completion. TLS and KCM use stream parsing.

- How do I attach my program? A redirection progaram is attached to a sockmap as BPF_SK_SKB_STREAM_VERDICT; it should return the result of bpf_sk_redirect_map(). A strparser program is attached via BPF_SK_SKB_STREAM_PARSER and should return the length of data parsed.

- What context is provided? A pointer to the struct __sk_buff containing packet metadata/data. However more fields are accessible to the sk_skb program type. The extra set of fields available are documented in include/linux/bpf.h like so:

```
 /* Accessed by BPF_PROG_TYPE_sk_skb types from here to ... */
  u32 family;
  u32 remote_ip4;   /* Stored in network byte order */
  u32 local_ip4;    /* Stored in network byte order */
  u32 remote_ip6[4];    /* Stored in network byte order */
  u32 local_ip6[4]; /* Stored in network byte order */
  u32 remote_port;  /* Stored in network byte order */
  u32 local_port;   /* stored in host byte order */
/* ... here. */
```

So from the above alone we can see we can gather information about the socket, since the above represents the key information that identifies the socket uniquely (protocol is already available in the globally-accessible portion of the struct __sk_buff).

- When does it run? A stream parser can be attached to a socket via BPF_SK_SKB_STREAM_PARSER attachment to a sockmap, and the parser runs on socket receive via smap_parse_func_strparser() in kernel/bpf/sockmap.c . BPF_SK_SKB_STREAM_VERDICT also attaches to the sockmap, and is run via smap_verdict_func().